C O M P U T E R A R T

GO-88 - A DUAL MODE CPU BOARD FOR THE S-100 BUS

GO-88 MANUAL

by

A G Biggs

CMB 59
BABINDA QLD 4861

(070) 674124

JUNE 1985

Copyright (C) 1985 COMPUTER ART

CONTENTS

		Page
1.	Introduction	1
2.	Overview of GO-88	1
3.	Options	2
4.	Model coprocessor system	6
5.	Initial setting-up - Hardware	6
6.	Initial setting-up - Software	9
7.	Monitor commands	10
8.	Theory of operation	14
	8.1 Mode change	14
	8.2 Stand-alone mode	14
	8.3 Coprocessor mode	15
9.	Coprocessor programming	17
	9.1 Z80 programs calling 8088 subroutines	18
	9.2 8088 programs calling Z80 subroutines	19
	References	21
	Appendix A	
	MON88.COM source and object code	22
	Tables	
1.	GO-88 parts list	23
2.	S100 lines used by GO-88	25
	Figures	
1.	GO-88 circuit - sheet 1	27
2.	GO-88 circuit - sheet 2	28
3.	Memory map - model coprocessor system	29
4.	GO-88 bus swap sequence	30

1. INTRODUCTION

The purpose of this manual is to provide the GO-88 owner with sufficient information to set up the hardware and coprocessor software and get it going in his system to the stage where the monitor commands can be executed. Instructions for setting up and running CPM86 are covered by a supplement and a diskette. The theory of operation of GO-88 is explained in sufficient depth to enable troubleshooting.

Before inserting GO-88 into the S100 system the purchaser should read this manual, paying particular attention to sections 5 and 6, Initial setting-up - Hardware and Initial setting-up - Software.

2.OVERVIEW OF GO-88

This versatile board has two basic modes of operation:

1. as an asynchronous coprocessor board using INTEL's powerful 8088 chip (reference 1) on the same S-100 bus with any IEEE696* compatible 8080/85/Z80 master CPU board.

2. as a stand-alone 8088 CPU board compatible with or exceeding IEEE696 requirements.

Changeover between these two basic hardware modes is achieved by removing one TTL chip from its socket and replacing it with a specially-wired DIP plug.

In the coprocessor mode changeover between your existing CPU board and GO-88 is entirely under software control. COMPUTER ART has developed a fast and reliable bus exchange mechanism which, despite its asynchronous nature, meets the new IEEE696 bus exchange protocol. GO-88 has been extensively tested at many different master and coprocessor clock frequencies and in our development computer is used routinely with an 8MHz 8088-2 and a 5.5MHz Z80B without wait states. This capability has several important advantages:

(i) each processor can operate at its maximum speed or at the maximum speed bus slaves will allow. For example, an 8088 can operate at 4.5MHz with standard 450ns memory, whereas a Z80 can only operate at 2MHz

(ii) one can mix Z80 and 8088 code in the same program, thus bringing 16-bit processing power to an existing 8-bit computer. In general, this capability can be used in two ways:

   (a) Z80 programs calling 8088 subroutines. For example, fast 16-bit multiply and divide. COMPUTER ART has developed patches for several high-level languages. These will be available from COMPUTER ART on floppy disk together with the GO-88 monitor listing.

   (b) 8088 programs calling Z80 subroutines - for example, the monitor in EPROM for GO-88 knows how to make CPM 2.2 BDOS calls, so all the facilities of CPM 2.2 are at the disposal of GO-88. This means one can develop and run 16-bit programs on an existing 8-bit operating system.

* IEEE696 is the new standard for the S100 bus (references 2,3)

(iii) an entire 16-bit operating system can run on an existing 8-bit computer. COMPUTER ART has developed a BIOS for GO-88 which enables full CPM86 operation in the coprocessor mode with a Morrow DISCUS 2D controller.

GO-88 has extended (24-bit) addressing and can directly address a megabyte of memory. To enable it to be used with existing 16-bit addressed memory boards GO-88 optionally generates PHANTOM to disable these boards except when they are addressed in the 64K chapter of memory to which they are allocated.

While there are several 16-bit CPU boards on the market, they require either the complete scrapping of existing 8-bit software, and most hardware as well, or as a part of the package a different 8-bit processor which may or may not be compatible with existing Z80 software. With GO-88 your entire hardware and software investment is preserved, a rare event in the computer industry these days. You can move into 16-bit computing as gradually or as fast as you wish.

Because few S-100 systems completely conform to the IEEE696 standard GO-88 has a number of options to help overcome such problems. Others can be overcome by simple modifications to existing boards, some of which would be necessary anyway, in any upgrading of the system. We are building up a library of modifications for popular boards.

GO-88 is based on the INTEL 8088 8284A chip set (reference 1). Internally the 8088 is a 16-bit processor but it communicates with the outside world through an 8-bit data bus. The 8284A is the clock generator and uses a crystal three times the clock frequency, i.e. a 5 MHz clock requires a 15 MHz crystal. The 8088 is configured in the MIN mode, but the board was designed so that it could be altered to the MAX mode, although this would be no advantage unless it was desired to include an 8087 number cruncher or an 8089 I/O processor. Pads for an extra 40-pin socket are included for such an eventuality. The board provides no I/O ports as in most applications these would already be available in the system. The spare 40-pin pad could be used for a PPI.

The board has two 5-volt regulators so there is plenty of spare 5-volt power. Power required is 800ma at 8 volts for 8MHz operation, or 700ma at 5MHz.

The on-board memory consists of a 4K byte 2732 EPROM and 6116 2K CMOS RAM. The standard monitor occupies less than 2K of the EPROM and uses the 6116 as scratchpad RAM.

All signals to and from the board are buffered to IEEE696 standards.

The printed-circuit board is of high quality double-sided construction on fibreglass with plated-through holes, silk-screened and solder masked, with gold-plated S100 contacts. The traces are wider than usual and this, with liberal by-passing, results in exceptionally clean and fast signals. Table 1 is a complete list of components and figures 1 and 2 provide all circuits.

3. OPTIONS

This section provides an overview of GO-88's hardware and software options. Their specific uses are discussed in sections 5 and 6.

(a) Dual mode

One of the unique features of GO-88 is its dual personality. It can work

either as a coprocessor on the S100 bus sharing the processing tasks with your existing Z80, or whatever, or it can be a stand-alone CPU card (bus master in IEEE696 parlance). This transformation is achieved by removing one integrated circuit (IC16 - a triple 3 input AND gate) from its 14-pin socket and replacing it with the DIP plug provided. This activates on board logic that "rewires" the board and saves having lots of switches and/or solder links. The standard monitor program provided in the 2732 EPROM for the coprocessor mode will not work in the stand-alone mode and has to be replaced with one programed for your system.

(b) Memory management

Most of the memory boards used in existing S100 systems have 16-bit addressing, i.e. their addressing range is 0 to 64K or 0-FFFF hexadecimal. For them, the world ends at FFFFH (or rather starts again). If you throw a 23-bit address at these boards they recognize only the lower 16 bits and so they come up in every 64K chapter of the 8088's address range. One way out of this dilemma is to use "bank select" but that is cumbersome. GO-88 has an on-board memory manager to take care of it. Put simply, you decide which chapter of the 8088's address space you want the 16-bit addressed boards to belong to, i.e. 00000 to 0FFFFH, or 10000 to 1FFFFH, or ....., and then set 4 binary coded DIP switches (for chapters 0 to 15) to the chapter number. It works by asserting PHANTOM (S100/67) whenever the boards are not required. Of course your memory boards have to be able to respond to PHANTOM. This facility can be used in either the stand-alone or the coprocessor mode.

(c) Processor exchange

Another unique feature of GO-88 is its asynchronous bus swap facility. For two processors to time share a common bus under program control you have to have a fast and reliable bus swap mechanism. Unfortunately the S-100 bus was not well thought-out in this regard because inevitably the process requires floating (tri-stating) some of the bus signals during the swap. The catch is that some of the CONTROL bus signals that cause READ's and WRITE's to memory, etc. float to the ACTIVE state! To ensure that this never happens both processors have to drive the CONTROL bus together in the inactive state, for a brief interval of time. It is like relay runners passing the baton. If they throw it eventually someone will drop it, with disastrous results. For a brief interval both runners must have hold of the baton. This is explained in more detail in section 8.

The IEEE696 committee established a protocol for bus swapping and GO-88 follows that protocol. They also established a hardware arbitration procedure (which takes lots of chips) in case multiple "temporary masters" should simultaneously request the bus. This could be the case with input/output controllers wanting to transfer data at the whim of peripherals. GO-88 does not implement hardware arbitration because the master always passes the bus to it under program control so, if necessary, the master can resolve potential conflicts by a software instruction. Nested TMA operations are not permitted under the protocol.

To understand the foregoing you really need to read the IEEE696 report (references 2,3). Program-wise, you exchange processors as follows:

if Z80 has the bus it executes an OUT (GO88),A

if 8088 has the bus it executes an OUT (GO80),AL

where GO80 is an even numbered port between 0 and 255
and GO88 = GO80+1; A & AL can have any value.

You select the required port with 7 binary coded DIP switches (S1) on GO-88. The standard EPROM and software assumes GO80=20H and GO88=21H.

Before we leave this topic, note that it is absolutely essential that your existing CPU board can respond to HOLD on S100/74 and return PHLDA on S100/26. Also, it must be able to float its address, ADSB (S100/22), status SDSB (S100/18), data out DODSB (S100/23) and control CDSB (S100/19) buses, the latter separately. Most boards can do this (there might be problems with SBC-200). These are all, except PHLDA, open collector lines and GO-88 provides 1.5K pull-up resistors on each.

(d) Reset and Power-on logic

Strictly in accordance with IEEE696. Some CPU boards (incorrectly) generate POC (S100/99) every time RESET (S100/75) is activated, resulting in the system locking into the RESET state when GO-88 is plugged in (coprocessor mode). A simple fix for this problem is detailed in section 5.

(e) MWRT

MWRT (S100/68) can be generated anywhere in the system but wherever it is generated it must float to the inactive (low) state during bus exchanges. GO-88 provides correct MWRT generation as an option, for both the stand-alone and coprocessor modes.

(f) PSTVAL

Processor Status Valid (S100/25) which used to be known as Ø1 is a CONTROL bus signal and therefore should be floated when CDSB is active. GO-88 does this correctly but most boards do not. To avoid contention GO-88 provides a link so that PSTVAL can be disabled. The problem only arises in the coprocessor mode and in that mode GO-88 needs only to access system RAM.

(g) 2MHz CLOCK

GO-88 provides the mandatory 2MHz CLOCK on S100/49. To avoid contention on-board logic automatically disables it in the coprocessor mode.

(h) 2716 EPROM instead of 6116 RAM

A link is provided so that a 2716 EPROM can be substituted for the 6116 RAM. That would provide a total of 6K bytes of on-board EPROM. If more is needed the spare 40 pin pads could accommodate any size of EPROM.

(i) Pads for extra 40-pin socket

Some possible uses for this have already been mentioned.

(j) NMI

A link is provided to disable Non-Maskable Interrupt, on S100/12.

(K) Unterminated buses

For the on-board memory manager to correctly manage 16 bit addressed boards in the stand-alone mode, it is necessary that the extended address lines A16 to A23 when floated for TMA operations float to the high state. With terminated buses this will happen anyway. Pads are provided for extra pull-up resistors to be installed when the board is used on unterminated buses. These should be installed only if GO-88 is used on an unterminated bus.

(l) Software

The standard software provided with GO-88 has been designed to work in the coprocessor mode in a wide variety of CPM80 systems. By CPM80 we mean CPM1.4 to 2.2. It is not possible to provide standard software for the stand-alone mode because there are so many different host system configurations. The standard package contains the following:

(i) monitor in the 2732 EPROM

(ii) listing of the initialization routine MON88.COM

Both are available as listings on an 8" SD diskette for a small extra charge.

The monitor contains the following commands:

B stand-alone mode: load and run operating system
C coprocessor mode: call any CPM80 BDOS function (includes all disk functions)
C stand-alone mode: (for DJ-2D) call any subroutine with FAR RETURN
D dump any memory in hex. and ASCII
E enter data into memory in hex. or ASCII
F fill memory with hex. or ASCII data
G run 8088 program at any address with breakpoints
I input data via 8088 input port
M block move data intra or inter-segment, up, down, overlapping
O output data via 8088 output port
R display and alter 8088 registers and flags
S search memory for hex. or ASCII sequence
T trace 8088 program one or n steps displaying registers.
V verify that two blocks of memory are identical and display any differences.

There is space in the monitor EPROM for user programs.

Note that in the coprocessor mode the monitor requires a small area of memory within the CPM80 address space which will not be overwritten by programs or the

operating system. This could be anywhere within the 64K but the standard EPROM assumes the following:

003BH to 003FH inclusive; 5 bytes

It is not used by CPM80, nor by any application programs that we are aware of. If the area is not available the EPROM would have to be reprogramed and MON88.COM changed. COMPUTER ART can do this for a small extra charge.

4. MODEL COPROCESSOR SYSTEM

At this stage it is useful to introduce the concept of a "model" system. As a coprocessor, GO-88 adds completely new dimensions to an existing 8 bit computer. Many new configurations are possible. Ideally, as the system expands, all hardware and software will be upward compatible with the earlier system. In order to give direction to such development we have defined a model system. At the very least the model system would fully implement the CPM86 operating system. If that is achieved any CPM86 program will run in the system. At the same time this concept has enabled us to design the GO-88 monitor program so that it will work without modification in any system from the smallest CPM80 system to a fully expanded CPM86 system and beyond.

There are some potential areas of conflict in running an 8088 in a CPM80 system. INTEL designed the 8088 so that its first 1K of memory address space is reserved for interrupt vectors. Unlike Z80's etc., the 8088 allows software interrupts and both CPM86 and our monitor use them. CPM80 reserves much of the first 256 bytes of memory for system purposes. Our Version 1.0 implementation of CPM86 avoided this conflict by allocating chapter 1 of memory, not chapter zero, to the CPM80 system and loaded CPM86 also into chapter 1. This caused hardware difficulties in some systems; our current software works with CPM80 in chapter 0 where it is more natural for it to be, and CPM86 also loads into chapter 0. Conflicts are avoided by saving and restoring any memory locations that would otherwise clash. This has included consideration of the requirements of CPM80,the GO-88 monitor,CPM86 and MSDOS. Figure 3 is a memory map of the model coprocessor system.

Note that this assumes the floppy disk controller is memory mapped i.e., it occupies memory starting at 0E000H. There would be more memory available to both CPM80 and CPM86 if the controller were input/output mapped. Note also there are two separate CPM86 TPA's (Transient Program Areas -where user programs are loaded). This is no problem because CPM86 can manage up to 8 non-contiguous areas. If you have memory boards with 16 bit addressing (such as the Morrow Discus 2D floppy disk controller) they should all be in chapter 0 of memory with PHANTOM enabled so that GO-88 can manage them(see section 5 (1).

5. INITIAL SETTING-UP - HARDWARE

When you have unpacked GO-88 you should check the following:

(a) Locate the 14 pin DIP plug provided to replace IC16 for stand-alone operation. If the board is to be used in the stand-alone mode carefully remove IC16, prising it out with a small screwdriver, or use an IC extractor, and insert the DIP plug with its dot or indentation towards the centre of the board. Either the DIP plug or the 74LS11 should be put away for possible

future use.

(b) Examine the board carefully for signs of damage. If it is damaged, return it immediately to COMPUTER ART. Check that all IC's are pressed firmly in their sockets.

(c) Check the monitor EPROM (IC22). If it is the standard EPROM, intended for use in the coprocessor mode, it will be labelled MON88C. A stand-alone monitor would be labelled MON88S.

(d) Check whether your S100 mother board is actively terminated or not. If it is actively terminated all signal lines except possibly S100/75 RESET will be pulled-up to 2.6 volt + or - 0.2 volt via a 180ohm resistor, in which case only the first three resistors in the SIP at R6 will be used and SIP R12 will not be installed. If your bus is not actively terminated, i.e. it floats, 7x1.5K or 7x2.2K SIP resistors should be installed at both R12 and R6.

(e) Resistor R19 (1.5K) should be installed only if there is no provision on other boards or anywhere else in the system for S100/75 (RESET) to be pulled up high when the system RESET button is released.

(f) A link should be installed between holes 1 and 2 (just above IC23) unless a 2716 EPROM is to be installed at IC23 in which case only 2 and 3 should be linked.

(g) Read section 3(e) concerning MWRT. A link between holes 7 and 8 (just to the left of IC34) will enable GO-88 to generate MWRT for the whole system. If MWRT (S100/68) is already being correctly generated elsewhere this link should not be installed. By "correctly generated" we mean that MWRT should be the result of NOR'ing SOUT (S100/45) and PWR (S100/77), so that MWRT will go inactive (LOW) if SOUT or PWR, or both are floated. Of the Australian CPU boards ZERO 1 does it exactly right, but Applied Technology DG-680 Rev. B boards are wrong. With the latter, either cut DG-680's track to S100/68 and let GO-88 generate MWRT for the system or modify the DG680 board.

(h) Read section 3(f) concerning PSTVAL. Since few S100 CPU boards float S100/25, to avoid contention on this line it is safer not to insert the link between holes 9 and 10. We do not know of any memory boards that require PSTVAL but if such exist they should work with GO-88 in the stand-alone mode (provided 9 and 10 are linked) but not in the coprocessor mode. Incidentally, S100/25 provides a sneaky way of getting the inverse of the 8088 clock onto the bus in the coprocessor mode, provided the master can float its version.

(i) If Non-Maskable Interrupt is required a link should be placed between holes 11 and 12. Note that the standard software does not support NMI.

(j) Check whether your CPU board generates POC (S100/99) and RESET (S100/75) according to IEEE696. If it is correct POC will only become active when the system power is switched on; in particular, POC should NOT be activated every time the RESET button is pressed, i.e. POC should generate RESET but not RESET generate POC. If your system is in accordance with IEEE696 GO-88 will accept these signals from the master CPU without modification. If your system generates POC from RESET, GO-88 will simply lock the system into the RESET mode. The easiest way out of this impasse is to remove IC8 from its socket and carefully bend out pins 11 and 13 so that when IC8 is reinserted, pins 11 and

13 do not make contact. Remember, pins are numbered clockwise from the indentation or dot end when viewed on the underside of the IC. Remember also to straighten these pins if the board is used in the stand-alone mode.

(k) Set the DIP switches S1 to select the correct output port for bus swapping in the coprocessor mode (see section 3(c)). They are binary coded and each switch when ON has the following decimal weight:

Switch number (address line select)	7	6	5	4	3	2	1
Weight (decimal)	128	64	32	16	8	4	2

You can choose only an even numbered port to select your master board; GO-88 automatically responds to the next (odd) number port to select itself. The standard software assumes GO80 = 20 hex, i.e. 32 decimal, so only switch 5 of S1 should be ON.

(l) Now for memory management, which is controlled by DIP switches S2. Irrespective of the size of your system, your 16 bit addressed memory boards should be allocated to chapter 0 of the 8088's address space. The standard software assumes that CPM80 also functions in chapter 0. The silk-screened legend above DIP switch S2 identifies switches 1 to 4 as corresponding to address lines 16 to 19. They are binary coded and each switch when ON has the following decimal weight:

Switch number	4	3	2	1
Address line	19	18	17	16
Weight (decimal)	8	4	2	1

For example, if all four switches were ON, their weight adds up to 15, or F hex, so the 16 bit addressed boards would be allocated to chapter F hex. of memory, i.e. F0000H to FFFFFH. If all four switches were OFF, their weight adds up to 0, so the 16 bit boards would be allocated to chapter 0, i.e. 00000H to 0FFFFH. So all four switches should be OFF. The switch marked PHANTOM, switch 6 of S2, should be ON unless you want to disable the memory management capability. Now you must make sure that each of your 16 bit addressed memory boards has PHANTOM enabled. If you have no 16 bit addressed boards in your system, that is to say all boards have extended addressing (20 or 24 bit) you should disable memory management by having switch 6 of S2 OFF. The extended address boards should be allocated to memory in accordance with their manufacturer's instructions. If you have both 16 bit addressed and extended address boards in your system the extended address boards must NOT have PHANTOM enabled. Note that some RAM boards have PHANTOM permanently enabled. For example, to disable PHANTOM on COMPUPRO's RAM 16 boards you have to cut the track to S100/67. Note also that some 16 bit addressed RAM boards provide PHANTOM only for READ's and do not PHANTOM on WRITE. This is non-IEEE696 and GO-88's memory management will not work with these boards unless they are modified to PHANTOM on both READ's and WRITE's. Digital Research Computers of Texas' popular 16K RAM boards are in this category. To make them PHANTOM on both READ and WRITE do NOT insert a link between the plate-through holes marked PHANTOM. Instead, remove IC47 from its socket, bend out pin 11 and solder a short length of wire-wrap between the bent-out pin and the right hand PHANTOM hole (the one that goes to S100/67). While you have the Digital Research board out check the earthing of bypass capacitors C8 and C9. You will probably find they go nowhere. This produces obscure data errors. The remedy is obvious. Referring to GO-88 again, for memory management to work in the coprocessor mode GO-88 requires a link to be inserted between holes 5 and 6 just below IC26.

The board was designed so that when it is configured in the stand-alone mode (DIP plug in IC16 socket) the memory management facility is automatically disabled. If you still want to manage 16 bit addressed boards in that mode the link below IC26 should be connected between holes 4 and 6. The S2 DIP switches are switched as above.

(m) Table 2 contains a list of the S100 lines used by GO-88. It would be as well to check these against their usage in your existing system to make sure there are no remaining conflicts or incompatibilities due to non-IEEE696 usage.

(n) Having completed the above checks and selected the required options the board may be plugged into the bus. The power must be off whenever an S100 board is inserted or withdrawn.

6. INITIAL SETTING-UP - SOFTWARE

In the coprocessor mode the GO-88 monitor communicates with the CPM80 operating system through a common area of system memory (see section 3(1). There are actually two small areas in chapter zero of memory with addresses from 0003BH to 0003FH and from 00104H to 00119H for the standard software. These areas have to be initialized and that is the purpose of MON88.COM.

The computer boots up under CPM80 in exactly the same way as it does without GO-88 installed. Until an OUT (G088),A is executed by the master GO-88 remains dormant in the RESET condition. When the master executes this instruction GO-88 asserts HOLD; on receiving HOLD the master completes execution of the OUT instruction and issues PHLDA (hold acknowledge). The master then goes into a state of suspended animation. When GO-88 receives PHLDA the 8088 comes out of RESET and jumps to FFFF0H where the EPROM provides an inter-segment jump to 0003BH. One of the jobs MON88.COM does is to initialize a jump at that address taking the 8088 to the monitor program starting at FF000H (the first address in the EPROM). MON88.COM also sets up a loop at 00104H within which the Z80 will circulate and from which it is sent on its errands to the BDOS by GO-88. A listing of the source program MON88.Z80 and the hexadecimal object code is in Appendix A. A study of the listing and its comments will explain exactly how the above is done.

MON88.Z80 (or an ASM version with 8080 mnemonics) can be assembled to produce MON88.COM or, alternatively, the hexadecimal code can be keyed in with DDT and SAVED as MON88.COM. Check that the object code is identical with Appendix A.

If all has been done correctly, entering MON88 from any disk should result in:

```
GO-88 Monitor
coprocessor mode - CPM in Ch.0  V1.1
>(prompt)
```

The monitor is ready to receive commands. A list of the commands and their syntax is in the next section.

7. MONITOR COMMANDS

Control-H backspaces and deletes.

Control-S starts and stops scroll;
stand-alone mode only: any other character cancels command.

Control-X cancels the command line.

Control-P starts and stops printer in coprocessor mode only.

A blank or comma is the general delimiter.

Numbers are always entered as hexadecimal numbers.

Letters are only upper case unless in ASCII strings.

ASCII strings are delimited with single or double quotes.

Except where indicated otherwise, addresses are entered as absolute addresses with from one to five hexadecimal digits. For examples, the address of the start of the on-board scratch-pad RAM is FE800 while the CPM80 system starts at 0.

C 0 0 exits the monitor and returns to CPM80. This is a special case of the C command.

B Stand-alone mode: boot (i.e. load) and run CPM86

C Coprocessor mode: call any CPM80 BDOS function.

Syntax:

C <1 or 2 hex digit CPM funct.> <1 to 4 hex digit parameter>

See CPM80 documentation for a list of CPM function calls.
All 8088 registers are displayed. The result, if any, of the call will be in the 8088 register corresponding with the CPM80 register:

AL	corresponds	with	A
BX	"	"	HL
CX	"	"	BC
DX	"	"	DE

Where the parameter is an address it is limited to four hexadecimal digits because CPM80 recognizes no more.

C Stand-alone mode: call any subroutine with a FAR RETURN.

Syntax:

C

will execute subroutine at the address defined by the contents of the 8088 instruction pointer (IP) and code segment (CS) registers and return to the monitor displaying all registers.

D Display any memory in hexadecimal and ASCII.

Syntax:

D <start address>

will display 128 bytes from start address.

D <start address> <end address>

will display all memory between these addresses.

D <start address> <L 1 to 4 hex digits>

will display a length one to four hex digits long, starting at start address.

E Enter data into memory in hex or ASCII.

Syntax:

E <start address>

Enter data one hex byte at a time, stepping through memory using the space bar. Minus steps backwards. RETURN exits the command.

E <start address> <sequence of hex bytes>
or
E <start address> <'ASCII string'>

enters sequence of hex bytes or an ASCII string into memory from start address.

F Fill memory with hex or ASCII data.

Syntax:

F <start address> <end address> <sequence of hex bytes>
or
F <start address> <end address> <'ASCII string'>

repeats the sequence or ASCII string from start address to end address.

F <start address> <L 1 to 4 hex digits> <sequence of hex bytes>
or
F <start address> <L 1 to 4 hex digits> <'ASCII string'>

repeats the sequence or ASCII string from start address through length L.

G Run 8088 program at any address with breakpoints.

Syntax:

G <breakpoint 1> <breakpoint 2>.....<breakpoint 10>

execution will commence at the address defined by the contents of the 8088 instruction pointer (IP) and code segment (CS) registers. It will stop at the next breakpoint and the contents of all registers will be displayed and all breakpoints will be cleared.

I Input data via 8088 input port.

Syntax:

I <1 to 4 hex digit port address>

input and display data from the addressed port.

M Block move data intra or inter-segment, up, down, overlapping.

Syntax:

M <source start address> <source end address> <destination start address>

move the block of data between start address and end address to destination start address.

M <source start address> <L 1 to 4 hex digits> <destination start address>

move block of data of length L from source start address to destination start address.

O Output data via 8088 output port.

Syntax:

O <1 to 4 hex digit port address> <hex byte>

output a byte to the addressed port.

R Display and alter 8088 registers and flags.

Syntax:

R

display all registers and flags.

R <16 bit register name>

display and alter a particular register.

R F

displays all flags and prompts to alter a particular flag using the

following mnemonics:

Flag	Set	Reset
Overflow	OV	NV
Direction	DN	UP
Interrupt enable	EI	DI
Sign	NG	PL
Zero	ZR	NZ
Auxiliary carry	AC	NA
Parity	PE	PO
Carry	CY	NC

S Search memory for a hex or ASCII sequence.

Syntax:

S <start address> <end address> <hex byte sequence>
or
S <start address> <end address> <'ASCII string'>

searches memory between start and end address and displays the start address of each occurrence of the byte sequence or ASCII string.

S <start address> <L 1 to 4 hex digits> <hex byte sequence>
or
S <start address> <L 1 to 4 hex digits> <'ASCII string'>

searches L bytes of memory from start address and displays start address of each occurrence of the byte sequence or ASCII string.

T Trace 8088 program one or more steps displaying registers.

Syntax:

T <1 to 4 hex digits>

as in G command except program will stop after the nominated number of steps and display all registers. RETURN resumes execution.
T

as above except trace only one step.

V Verify that two blocks of memory are identical and display any differences.

Syntax:

V <block 1 start address> <block 1 end address> <block 2 start address>

verify that block 1 and block 2 are identical and display any differences.

V <block 1 start address> <L 1 to 4 hex digits> <block 2 start address>

verify that block 1 of length L is identical with block 2 of the same length

8. THEORY OF OPERATION

The aim of this section is to provide essential background on how GO-88 operates to facilitate fault diagnosis. It is assumed that the reader has access to reference 1 which explain how the 8088/8284A chip set works. Quite obviously, to try to understand GO-88 without having some knowledge of how the INTEL chips work is not likely to succeed.

The following explanations will also require constant reference to the circuits in figures 1 and 2.

8.1 Mode Change

When the IC16 socket contains a 74LS11 the board is in the coprocessor mode. When it contains a DIP plug wired as shown in figure 2 the board is in the stand-alone mode. The jumper between pins 1 and 12 of the DIP plug connects the buffered system RESET line to the $\overline{\text{RESET}}$ input of the 8284A. As pin 7 of the IC16 socket is grounded, the DIP plug also grounds pin 6 which makes CP (CoProcessor) false (low) and through IC33/2, 3 and 1 makes $\overline{\text{CP}}$ false (i.e. high). CP and $\overline{\text{CP}}$ are used to enable or disable a number of lines via tri-state buffers in IC29, IC31, IC32 and IC34. Note that this has the effect of changing the direction of some S100 lines; the changes are listed in Table 2 which is also a complete list of all S100 bus lines used by GO-88.

8.2 Stand-alone Mode

The DIP plug also grounds pin 8 of the IC16 socket so nothing can be clocked through to IC17/5 and 6 (the names of these signals, GO88 and GO80, have no meaning in the stand-alone mode). The removal of IC16 also disconnects the output of the bus swap port, IC5 and IC6. Following a system RESET IC17/5 and 6 will be permanently low and high respectively.

Because CP is low at IC12/12, IC12/11 will be permanently low. Thus, the flip-flops IC18 will clock permanent lows to IC12/5 and IC11/10. These two AND gates will therefore have permanently low outputs at IC12/6 and IC11/8. This means that the open collector outputs of IC9, pins 12, 6, 10 and 8 which are $\overline{\text{ADSB}}$, $\overline{\text{SDSB}}$, $\overline{\text{DODSB}}$ and $\overline{\text{CDSB}}$ respectively, are all pulled high (inactive) by their pull-up resistors and are configured as inputs to GO-88 via the IC31 buffers to which they are connected. The same applies to IC9/2 ($\overline{\text{HOLD}}$) because IC21/1 and 2 are both permanently low. Thus, in the stand-alone mode GO-88 responds to the standard S100 bus disable and $\overline{\text{HOLD}}$ signals.

Most of the rest of the circuitry in figure 1 follows fairly standard S100 practice and is self-explanatory. IC13 is a 3 to 8 decoder which provides the status signals via the transparent inverting latch IC1. Because the 8088 has multiplexed address and data buses, addresses have to be latched by the 8088's ALE pulse.

Turning to figure 2 the 8088 can drive only 20 address lines, so A20 to A23 are driven low whenever they are not floated.

IC27 and IC28 decode addresses to select the on-board RAM and EPROM respectively.

Because IC29/1 is low the 2MHz clock drives S100/49.

The power-up and reset circuitry needs some comment. Because IC29/13 is low the Schmitt trigger is enabled; R13 and C5 provide the necessary power-on-clear delay. Note that this delay must always be greater by some margin than the rise time of the bus 8V.D.C. supply. While $\overline{\text{POC}}$ is low both $\overline{\text{RESET}}$ and $\overline{\text{SLAVECLR}}$ are pulled low by IC8/13 and 11 respectively. See section 5(e) concerning R19.

The memory management facility is automatically disabled in the stand-alone mode because XFERII at IC26/1 and 2 is always low in that mode. This was done so that G0-88 would be completely standard IEEE696 in the stand-alone mode. If you want to use 16 bit addressed boards in your system with G0-88 in the stand-alone mode the memory management facility can be enabled by changing the link just below IC26 from holes 5 to 6 to holes 4 to 6. See section 5(l). The PHANTOM switch has to be ON.

8.3 Coprocessor Mode

For the most part G0-88 produces its signals in accordance with well-established S100 practice. The key to understanding its unusual features is the bus swap mechanism, so that will be explained first. It should be read in conjunction with IEEE Task 696.1/D2 - the proposed standard for the S100 bus - particularly section 2.8.2 on bus transfer protocol (references 2 and 3).

(a) Bus Swap - Master to G0-88

Suppose for a start G0-88 is in the coprocessor mode (IC16 in), the master is a Z80 card and it has control of the bus. The 8088 is dormant because, since power-on, RESET IC14/21 has been high (how RESET and power-on works will be described later). The DIP switches S1 on the inputs to XOR gates IC5 and IC6 are binary coded with their corresponding address lines to select any even address between 0 and 254 (decimal). For the standard software, port addresses 20H and 21H (hex) are used and the S1/5 switch is ON. If Z80 executes an OUT 21H,A instruction (A register can have any value) IC16 pins 9, 10 and 11 go high as well as A0. The RC filter following IC16/8 gives about 40ns noise immunity and then the high on D (IC17/2) is latched into Q asserting the request G088 for the coprocessor to take over the bus.

The only effect G088 has immediately is to assert S100/74 $\overline{\text{HOLD}}$. The Z80 master having received $\overline{\text{HOLD}}$ completes its OUT machine cycle, high Z's its address, data and control pins and asserts S100/26 (PHLDA). Going back, when G088 went high, G080 IC17/6 (permanent master request) went low. IC17/9 is high while the 8088 is in RESET and this ensures that XFERI and XFERII are low because IC11/9 and IC12/4 are both low. Note that figure 8 of the IEEE standard shows XFERI and XFERII as negative true. In our convention these signals are true when high. The effect of G080 going low is to make IC17/9 low immediately, and the low at IC19/2 and IC19/12 is clocked through by these two flip-flops, the latter on the next rising edge of Ø, as a high at IC11/9 (and half a cycle later as a high at IC12/4). Thus, the IC11 and 12 AND gates are prepared for their AND functions. Soon after, PHLDA makes IC18/2 and 12 high (IC12/12 is high). The rising edge of the master clock at IC18/3 clocks a high through to IC12/5 asserting XFERI and setting up the second half of IC18, so that

half a cycle later a high is clocked through to IC11/10 asserting XFERII.

XFERI floats the master's address, status and data-out buses and enables GO-88's control buffers, while XFERII floats the master's control bus. Thus, for the half a clock cycle between XFERI and XFERII both boards are driving the control bus.

During this period all of the master's control signals should be in their inactive state except PHLDA. As mentioned above, during this same period GO-88's control buffers (IC8) are enabled by XFERI, and IC21/11 is high, so GO-88's control outputs are also in the inactive state except PHLDA. Thus, there is no conflict in the two processor boards simultaneously driving the S100 control bus and it is never allowed to float free during the swap, eliminating the possibility of glitches or false strobes. Note that MWRT (S100/68), a particularly dangerous strobe, will float inactive only if it is generated according to the IEEE standard (section 2.7.5.3). For this reason the GO-88 board provides optional MWRT generation by means of link 7 to 8.

When XFERII goes high it disables via the open collector driver IC9/8, the master's control buffer (S100/19, $\overline{\text{CDSB}}$) so the S100 control bus now belongs completely to GO-88. XFERII also enables the latter's status, address and data-out buses (the S100 PHLDA remains high).

Looking at the AND gate IC16's inputs at pins 13, 1 and 2, pins 13 and 1 are already high and XFERII makes pin 2 high, thus lifting the 8088/8284's RESET; the CPU comes out of retirement and does its first instruction fetch from location FFFF0H where the on-board EPROM is programmed with an inter-segment jump to the desired code area (to 0003BH with the standard 2732).

(b) Bus Swap - GO-88 to Master

While GO-88 has control of the S100 bus XFERI and XFERII are both high. To return to the permanent master these two functions have to be returned to their inactive levels in reverse order in accordance with the IEEE protocol.

The bus exchange commences in similar fashion with the execution by the 8088 of an OUT N,AL instruction where N is the swap port switch setting, i.e. for the standard software an OUT 20H,AL (AL can have any value). This time A0 is low so IC17/5 (G088) is latched low and IC17/6 (G080) high. G088's going low has no immediate effect on S100/74 ($\overline{\text{HOLD}}$) because XFERI is still high and OR gate IC21 pins 1 and 3 latch $\overline{\text{HOLD}}$ true. However, the following events occur simultaneously: AND gate IC16/3 and 12 goes low activating the 8284A/8088 RESET logic and IC17/13 goes high preparing that flip-flop for clocking. The latter event has no immediate effect because the flip-flop has as yet nothing new to clock, as will now be explained.

On receipt of $\overline{\text{RESET}}$ low (IC15/11) the Intel clock generator waits internally, until the 8088 clock (IC15/8) goes low, to clock RESET high through to the 8088. The inverse of the 8088 clock, i.e. $\overline{\text{CLK}}$ at IC17/11 will not immediately clock RESET high through to IC17/9 because the 74LS74

requires a minimum of 20ns set-up time at its D input. It therefore waits one 8088 clock cycle to the next rising edge of CLK before changing IC17/9. Meanwhile the 8088 internally synchronizes RESET with the 8088 CLK (see Intel's 8086/88 manual, reference 1) so that on the next falling edge of CLK the 8088 drives its various output pins to their respective inactive states prior to high Z'ing those that float. At the same falling edge of CLK, the rising edge of CLK clocks the high state of the D input through IC17/9 and to the D inputs of IC19.

The next rising edge of the master clock (Ø) at IC19/3 clocks a low through to IC11/9 making XFERII low and at the same time prepares the second half of IC19 for clocking, half a master clock cycle later to de-activate XFERI.

XFERII's de-activation enables the master board's control buffers. At this stage all of the master's control outputs should be in the inactive state except PHLDA. At the same time GO-88's status, address and data-out buffers are disabled leaving only its control buffers driving the bus in parallel with the master's. Recall that the 8088 has already driven its control outputs to the inactive state so no bus conflict occurs. Thus, a smooth glitch-free transfer of the control bus to the master should ensure when XFERI goes low disabling GO-88's control buffers via IC32/10 and 9.

XFERI's de-activation also enables the address, status and data-out buffers of the master board and, via OR gate IC21/1 and 3, releases the HOLD on the master which in turn de-activates PHLDA (S100/26). The master recommences computing by executing an instruction fetch to the next instruction following the OUT 21H,A that disabled it.

Figure 4 shows the sequence of events. Note that for fault diagnosis in the coprocessor mode XFERII being high is a good indication that GO-88 has control of the bus. The link between holes 5 and 6 just below IC26 is XFERII and it has been shaped to allow easy connection of an oscilloscope or logic probe. When running the GO-88 monitor it will pulse on and off every time a character is output to the screen. While waiting for a keyboard character XFERII will be low because at that time the master CPU board will have control of the bus.

The reset and power-on logic is disabled in the coprocessor mode because IC29/13 is high allowing the bus master to perform these functions. At power-up, because the Schmitt trigger is tri-stated, R13 and C5 can never pull POC below the high level and so GO-88's RESET and SLAVECLR remain tri-stated. A problem arises however if, contrary to IEEE696, the master activates POC every time a RESET is executed. POC's going low will cause GO-88 to pull RESET low resulting in the system locking into the RESET mode. Either correct the master or modify GO-88 as described in section 5(j).

9. COPROCESSOR PROGRAMMING

The purpose of this section is to explain how assembly language programs may be written to use GO-88 in the coprocessor mode. It is assumed that the reader is familiar with at least one version of Z80/8080/8085 assembly language and understands Z80 mnemonics. It is not within the scope of this manual to teach the 8086/88 instruction set nor assembly language programming. The reader should become broadly familiar with these topics first by reading reference 1. Although the author has written short 8088 programs in machine code, be warned

that this is a much more tedious task than was the case with 8 bit processors. An 8086/88 assembler is almost essential and if you do not have CPM86 it would have to be a cross-assembler. COMPUTER ART is working on patches for high level 8 bit languages so that GO-88 will be called to do integer multiplication and division thereby speeding up programs and reducing their size. Thus, the user could obtain these advantages without doing any assembly language programming. The other route is to go straight to CPM86; COMPUTER ART is working on hybrid BIOS's to run CPM86 with popular 8 bit disk controllers. These have the advantage of being able to change from CPM80 to CPM86 or back again with a single command.

Coprocessor use may be divided into two categories: Z80 programs calling 8088 subroutines, and 8088 programs calling Z80 subroutines.

9.1 Z80 programs calling 8088 subroutines

The following piece of mixed Z80/8088 program could appear anywhere in a Z80 program. It multiplies two 16 bit signed factors, producing a 32 bit signed product.

```
GO80    EQU     20H
GO88    EQU     21H
ENTR88  EQU     003CH           ;address of 8088 initial jump.
SEG80   EQU     0000H           ;16 bit addresses start at 00000H
        .                       ; so they equal 8088 offsets.
        .
        .
;Assume address ENTR88-1 has been initialized with EAH (8088 JMP opcode)
; and ENTR88+2 with SEG80.
;Enter with
;FACTOR1 in DE
;FACTOR2 in HL
;
        LD      (FACTOR1),DE
        LD      (FACTOR2),HL
        LD      HL,MULTIPLY
        LD      (ENTR88),HL     ;tells 8088 where to jump.
        OUT     (GO88),A        ;freezes Z80 and wakes up 8088.
;8088 executes code at MULTIPLY and then Z80 resumes
;
        LD      DE,(PRODHI)     ;picks up high word
        LD      HL,(PRODLO)     ; and low word of product.
        RET
;
```

```
;8088 code follows
MULTIPLY:
        MOV     AX,CS
        MOV     DS,AX           ;initialize data segment register to 0000H
        MOV     AX,FACTOR1
        IMUL    FACTOR2         ;does the multiplication in 28us @ 5MHz
        MOV     PRODHI,DX       ;saves the product
        MOV     PRODLO,AX
        OUT     G080,AL         ;returns to Z80.
FACTOR1 DS      2
FACTOR2 DS      2
PRODLO  DS      2
PRODHI  DS      2
;more Z80 code follows
        .
        .
        .
```

In practice you use your cross-assembler to assemble the 8088 code first at any ORG to produce MULTIPLY.PRN just to find out how much space it requires. Then assemble the Z80 code reserving the required spaces (DS or DEFS) with the labels MULTIPLY,FACTOR1,FACTOR2,PRODLO and PRODHI in the correct places, producing (say) Z80PROG.HEX and Z80PROG.PRN. Now reassemble the 8088 code at the correct ORG and produce MULTIPLY.PRN and MULTIPLY.HEX. Next use DDT or ZSID to load Z80PROG.HEX and to overlay MULTIPLY.HEX. Finally exit ZSID and SAVE Z80/8088.COM or whatever. Use ZSID to check that all code is at correct address.

9.2 8088 programs calling Z80 subroutines

The example that follows is actually the character input routine used by GO-88's monitor contained in the 2732 EPROM. The monitor communicates with the outside world by means of CPM80 BDOS calls; to do this it sends the Z80 off on errands to the BDOS. The Z80 waits for its instructions at address SWAP (see Appendix A for the MON88 initialization routine where SWAP will be found).

The program below also contains a classic example of the incompatiblity of CPM80 low memory and the 8088 CPU; this is the reason why our early software loaded CPM80 into chapter 1 of memory and you can see how we have overcome that problem. In this case it arises because the 8088 does not like dividing something by zero. If you attempt this the 8088 will immediately generate a type 0 interrupt, that is, it will execute a CALL to the address contained in locations 00000 to 00003H. Now in CPM80 these locations contain the WBOOT jump and IOBYTE. For that reason the first thing our monitor does is to save the contents of those locations, expecting them to be overwritten by the user's interrupt servicing routine address. However the IN routine below calls CPM80 which expects to find IOBYTE at location 0003H, so the contents of the latter have to be replaced with IOBYTE and then restored.

```
G080    EQU     20H             ;address of bus swap port
CONIN   EQU     1               ;standard CPM80 BDOS function
SWAP    EQU     0104H           ;area of memory accessible by both CPU's
FUNC    EQU     SWAP+3          ;address where BDOS function is stored
RETBYT  EQU     SWAP+13H        ;address where returned byte is stored
ENTR88  EQU     003CH           ;address where inter-segment jump offset stored
SG80SAV EQU     17F3H           ;offset in GO-88 memory of EPROM location
                                ; containing CPM80 segment value,i.e. 00000H.
;Character input routine

IN:
        PUSHF                   ;RESET clears flags so save them
        PUSH    ES              ;RESET zeros ES so save its contents
        PUSH    DS              ; same with DS.
        PUSH    AX              ;save AX before we use it
        SEG     CS              ;segment overide prefix
        MOV     AL,[IOBYSAV]    ;get IOBYTE from GO-88 ram.
        SEG     CS
        MOV     DS,[SG80SAV]    ;CPM80 segment into DS
        XCHG    [0003],AL       ;restore IOBYTE, save interrupt byte
        MOV     W,[ENTR88],JUMP3;so that 8088 will know where to return.
        MOV     B,[FUNC],CONIN  ;desired BDOS function.
        OUTB    G080            ;resets 8088 and sends Z80 on errand to BDOS.

INRET:                          ;JUMP3 returns 8088 to here eventually.
        SEG     CS              ;segment overide prefix
        MOV     DS,[SG80SAV]    ;CPM80 segment into DS
        MOV     [0003],AL       ;restore interrupt byte
        MOV     AX,CS           ;CS was loaded by inter-segment jump
        MOV     SS,AX           ;  make SS the same.
        POP     AX              ;restore AX, AH might be important.
        MOV     AL,[RETBYT]     ;pick up returned byte from BDOS.
        AND     AL,7FH          ;strip off bit 7
        POP     DS              ;restore DS
        POP     ES              ;  and ES
        POPF                    ;  and flags.
        RET
        .
        .
        .
```

The mnemonic in the above program are those used by Seattle Computer Product's cross-assembler (reference 4). While the Z80 is going about its allotted task, the 8088 is in the RESET state. It might appear at first sight that the contents of those 8088 registers not saved would be lost in the RESET. We have found in fact that the contents of all 8088 registers except the flag and segment registers and of course the instruction pointer, are preserved while the 8088 is in the RESET state no matter how long. This saves a lot of PUSHing and POPing.

REFERENCES

1. iAPX 86,88 User's Manual. Intel Corporation. July,1981.

2. Standard Specification for S-100 Bus Interface Devices. IEEE Task 696.1/D2. COMPUTER. July,1979.

3. Libes and Garetz — Interfacing to S-100/IEEE696 Microcomputers. Osborne/McGraw-Hill. 1981.

4. Z80/8086 Cross Assembler. Seattle Computer Products,Inc. Seattle,WA. U.S.A.

APPENDIX A

MON88.COM SOURCE AND OBJECT CODE

```
;MON88.COM is the initialization routine for the GO-88 monitor in
;the coprocessor mode. You can either assemble the source listing
;below or just key in the hex code using DDT and SAVE it as MON88.COM.
;The 8088 always does its initial jump to FFFF0H in the EPROM which
;in turn contains a jump to (ENTR88-1) i.e. 0003BH.
;The code below initializes the jump at ENTR88, saves the BDOS address
;then jumps to the loop at SWAP where it awaits instructions from GO-88.
;Although ZILOG mnemonics are used, it will run on 8080/85 CPU's
;as well.


MONORG  EQU     0800H           ;GO-88 monitor origin.
MONSEG  EQU     0FE80H          ;monitor segment.
JMP88   EQU     0EAH            ;8088 inter-segment jump opcode.
GO88    EQU     21H             ;address of bus swap port.
BDOS    EQU     0005H           ;CPM80 BDOS entry.
ENTR88  EQU     003CH           ;8088 initial jump here.
        ORG     0100H
        JP      INIT
        NOP
SWAP:   OUT     (GO88),A        ;Z80 freeze here
        LD      C,0             ;Z80 back on job
FUNC    EQU     $-1             ;BDOS function from GO-88
        LD      DE,0
PARAM   EQU     $-2             ; also parameter
        CALL    0               ;CALL BDOS
SAVDOS  EQU     $-2             ; at saved address.
        LD      (RETBYT),A      ;in case BDOS answers back
        LD      (RETWD),HL
        JP      SWAP            ;Z80 back to freezer.
RETBYT  DS      1               ;GO-88 knows these addresses
RETWD   DS      2               ; and picks up the info.
INIT:   LD      HL,ENTR88-1
        LD      (HL),JMP88      ;initialize 8088 jump opcode
        INC     HL
        LD      BC,MONORG       ; and offset of monitor code
        LD      (HL),C
        INC     HL
        LD      (HL),B
        INC     HL
        LD      BC,MONSEG       ; and segment paragraph.
        LD      (HL),C
        INC     HL
        LD      (HL),B
        LD      HL,(BDOS+1)     ;saves BDOS address
        LD      (SAVDOS),HL
        JP      SWAP
        END
```

```
0100: C3 1A 01 00 D3 21 0E 00 11 00 00 CD 00 00 32 17
0110: 01 22 18 01 C3 04 01 00 00 00 21 3B 00 36 EA 23
0120: 01 00 08 21 23 70 23 01 80 FE 71 23 70 2A 06 00
0130: 22 0C 01 C3 04 01
```

TABLE 1 GO-88 PARTS LIST

LEGEND	COMPONENT	TYPE	FUNCTION
IC1	IC	74LS533	OCTAL INVERTING LATCH
IC2	IC	74LS244	OCTAL BUFFER
IC3	IC	74LS244	OCTAL BUFFER
IC4	IC	74LS244	OCTAL BUFFER
IC5	IC	74LS136	QUAD XOR OC
IC6	IC	74LS136	QUAD XOR OC
IC7	IC	74LS373	OCTAL LATCH
IC8	IC	74LS367	HEX BUFFER TRI-STATE
IC9	IC	7406	HEX INVERTER OC
IC10	IC	74LS04	HEX INVERTER
IC11	IC	74LS08	QUAD 2 INPUT AND
IC12	IC	74LS08	QUAD 2 INPUT AND
IC13	IC	74LS138	3 TO 8 DECODER
IC14	IC	P8088	CPU
IC15	IC	D8284A	8088 CLOCK GENERATOR
IC16	IC	74LS11	TRIPLE 3 INPUT AND
IC17	IC	74LS74	DUAL D FLIP FLOPS
IC18	IC	74LS74	DUAL D FLIP FLOPS
IC19	IC	74LS74	DUAL D FLIP FLOPS
IC20	IC	74LS04	HEX INVERTER
IC21	IC	74LS32	QUAD 2 INPUT OR
IC22	IC	2732	4K X 8 EPROM
IC23	IC	6116	2K X 8 STATIC RAM
IC24	IC	74LS373	OCTAL LATCH
IC25	IC	74LS125	QUAD TRI-STATE BUFFERS
IC26	IC	74LS11	TRIPLE 3 INPUT AND
IC27	IC	74LS30	8 INPUT AND
IC28	IC	74LS30	8 INPUT AND
IC29	IC	74LS125	QUAD TRI-STATE BUFFER
IC30	IC	74LS136	QUAD XOR OC
IC31	IC	74LS368	HEX INV TRI-STATE BUFFERS
IC32	IC	74LS368	HEX INV TRI-STATE BUFFERS
IC33	IC	74LS02	QUAD 2 INPUT NOR
IC34	IC	74LS125	QUAD TRI-STATE BUFFERS
IC35	IC	74LS00	QUAD 2 INPUT NAND
IC36	IC	7805	VOLTAGE REGULATOR
IC37	IC	7805	VOLTAGE REGULATOR
	PCB	GO-88	PRINTED CIRCUIT BOARD
C1	CAPACITOR	39PF	
C3	CAPACITOR	.01UF	
C4	CAPACITOR	0.1UF	BY-PASS
C5	TANT CAP	47UF 16V	BY-PASS
C6	TANT CAP	47UF 16V	BY-PASS
C7	TANT CAP	47UF 16V	BY-PASS
C8	TANT CAP	47UF 16V	BY-PASS
C9	TANT CAP	47UF 16V	BY-PASS
C10	CAPACITOR	0.1UF	BY-PASS
C11	CAPACITOR	0.1UF	BY-PASS
C12	CAPACITOR	0.1UF	BY-PASS
C13	CAPACITOR	0.1UF	BY-PASS
C14	CAPACITOR	0.1UF	BY-PASS

TABLE 1 GO-88 PARTS LIST

LEGEND	COMPONENT	TYPE	FUNCTION
C15	CAPACITOR	0.1UF	BY-PASS
C16	CAPACITOR	0.1UF	BY-PASS
C17	CAPACITOR	0.1UF	BY-PASS
C18	CAPACITOR	0.1UF	BY-PASS
C19	CAPACITOR	0.1UF	BY-PASS
C20	CAPACITOR	0.1UF	BY-PASS
C21	CAPACITOR	0.1UF	BY-PASS
C22	CAPACITOR	0.1UF	BY-PASS
C23	CAPACITOR	0.1UF	BY-PASS
C24	CAPACITOR	0.1UF	BY-PASS
C25	CAPACITOR	0.1UF	BY-PASS
C26	CAPACITOR	27PF	
R1	SIP RESIS	7 X 1.0K	PULL-UP
R2	RESISTOR	1K	PULL-UP
R3	RESISTOR	1K	
R4	RESISTOR	1.5K	PULL-UP
R5	RESISTOR	1.5K	PULL-UP
R6	SIP RESIS	5 X 2.2K	PULL-UP
R7	RESISTOR	1.5K	PULL-UP
R8	RESISTOR	1.5K	PULL-UP
R9	RESISTOR	1.5K	PULL-UP
R10	RESISTOR	1.5K	PULL-UP
R11	SIP RESIS	7 X 1.0K	PULL-UP
R13	RESISTOR	22K	
R14	RESISTOR	270	
R15	RESISTOR	560	
R16	RESISTOR	2.7K	
R17	RESISTOR	560	
R18	RESISTOR	1.5K	PULL-UP
R	RESISTOR	1.5K	PULL-UP
R20	RESISTOR	1.5K	PULL-UP
S1	DIP SWIT	8 POLE	
S2	DIP SWIT	8 POLE	
X1	CRYSTAL	15MHZ	CPU CLOCK
X2	CRYSTAL	2.0MHZ	2.0MHZ CLOCK
	HEAT SINK	2 X 7805	
	IC SOCKET	1 X 40P	
	IC SOCKET	2 X 24P	
	IC SOCKET	6 X 20P	
	IC SOCKET	1 X 18P	
	IC SOCKET	4 X 16P	
	IC SOCKET	21 X 14P	
	DIP PLUG	14P SOLD.	MODE CHANGE

TABLE 2 S100 LINES USED BY GO-88

PIN NO	NMENONIC	DESCRIPTION	ACTIVE LEVEL	DRIVER TYPE	DIRECTION STAND ALONE	DIRECTION CO-PROCESSOR
1	+8V	POWER				
3	XRDY	AUXILIARY READY	H		I	I
12	NMI	NON-MASKABLE INTERPT	L	OC	I	I
15	A18	ADDRESS BIT 18	H		O	O
16	A16	ADDRESS BIT 16	H		O	O
17	A17	ADDRESS BIT 17	H		O	O
18	SDSB	STATUS BUS DISABLE	L	OC	I	O
19	CDSB	CONTROL BUS DISABLE	L	OC	I	O
22	ADSB	ADDRESS BUS DISABLE	L	OC	I	O
23	DODSB	DATA OUT BUS DISABLE	L	OC	I	O
24	0	SYSTEM CLOCK	H		O	I
25	PSTVAL	STATUS VALID STROBE	L		O	O
26	PHLDA	HOLD ACKNLGE STROBE	H		O	I
29	A5	ADDRESS BIT 5	H		O	BI
30	A4	ADDRESS BIT 4	H		O	BI
31	A3	ADDRESS BIT 3	H		O	BI
32	A15	ADDRESS BIT 15	H		O	O
33	A12	ADDRESS BIT 12	H		O	O
34	A9	ADDRESS BIT 9	H		O	O
35	DO1	DATA OUT BIT 1	H		O	O
36	DO0	DATA OUT BIT 0	H		O	O
37	A10	ADDRESS BIT 10	H		O	O
38	DO4	DATA OUT BIT 4	H		O	O
39	DO5	DATA OUT BIT 5	H		O	O
40	DO6	DATA OUT BIT 6	H		O	O
41	DI2	DATA IN BIT 2	H		I	I
42	DI3	DATA IN BIT 3	H		I	I
43	DI7	DATA IN BIT 7	H		I	I
44	SM1	STATUS OP-CODE FETCH	H		O	O
45	SOUT	STATUS OUTPUT PORT	H		O	O
46	SINP	STATUS INPUT PORT	H		O	O
47	SMEMR	STATUS MEMORY READ	H		O	O
48	SHLTA	STATUS HALT ACKNOWLG	H		O	O
49	CLOCK	2.0MHz CLOCK	H		O	
50	GND	SYSTEM GROUND				
51	+8V	SYSTEM POWER				
54	SLAVECL	RESET BUS SLAVES	L	OC	O	
58	SXTRQ	STATUS 16 BIT REQST	L		O	O
59	A19	ADDRESS BIT 19	H		O	O
61	A20	ADDRESS BIT 20	H		O	O
62	A21	ADDRESS BIT 21	H		O	O
63	A22	ADDRESS BIT 22	H		O	O
64	A23	ADDRESS BIT 23	H		O	O
67	PHANTOM	DISABLES SLAVES	L	OC	O	O
68	MWRT	MEMORY WRITE STROBE	H		O	O
72	RDY	READY	H	OC	I	I
73	INT	INTERRUPT REQUEST	L	OC	I	I
74	HOLD	HOLD	L	OC	I	O
75	RESET	SYSTEM RESET	L	OC	BI	I

TABLE 2 S100 LINES USED BY G0-88

PIN NO	NMENONIC	DESCRIPTION	ACTIVE LEVEL	DRIVER TYPE	DIRECTION STAND ALONE	DIRECTION CO-PROCESSOR
76	PSYNC	BUS STATE 1 STROBE	H		O	O
77	PWR	WRITE STROBE	L		O	O
78	PDBIN	READ STROBE	H		O	O
79	A0	ADDRESS BIT 0	H		O	BI
80	A1	ADDRESS BIT 1	H		O	BI
81	A2	ADDRESS BIT 2	H		O	BI
82	A6	ADDRESS BIT 6	H		O	BI
83	A7	ADDRESS BIT 7	H		O	BI
84	A8	ADDRESS BIT 8	H		O	O
85	A13	ADDRESS BIT 13	H		O	O
86	A14	ADDRESS BIT 14	H		O	O
87	A11	ADDRESS BIT 11	H		O	O
88	DO2	DATA OUT BIT 2	H		O	O
89	DO3	DATA OUT BIT 3	H		O	O
90	DO7	DATA OUT BIT 7	H		O	O
91	DI4	DATA IN BIT 4	H		I	I
92	DI5	DATA IN BIT 5	H		I	I
93	DI6	DATA IN BIT 6	H		I	I
94	DI1	DATA IN BIT 1	H		I	I
95	DI0	DATA IN BIT 0	H		I	I
96	SINTA	STATUS INTER ACKNLGE	H		O	O
97	SWO	STATUS WRITE	L		O	O
99	POC	SYSTEM POWER ON CLR	L		O	I
100	GND	SYSTEM GROUND				

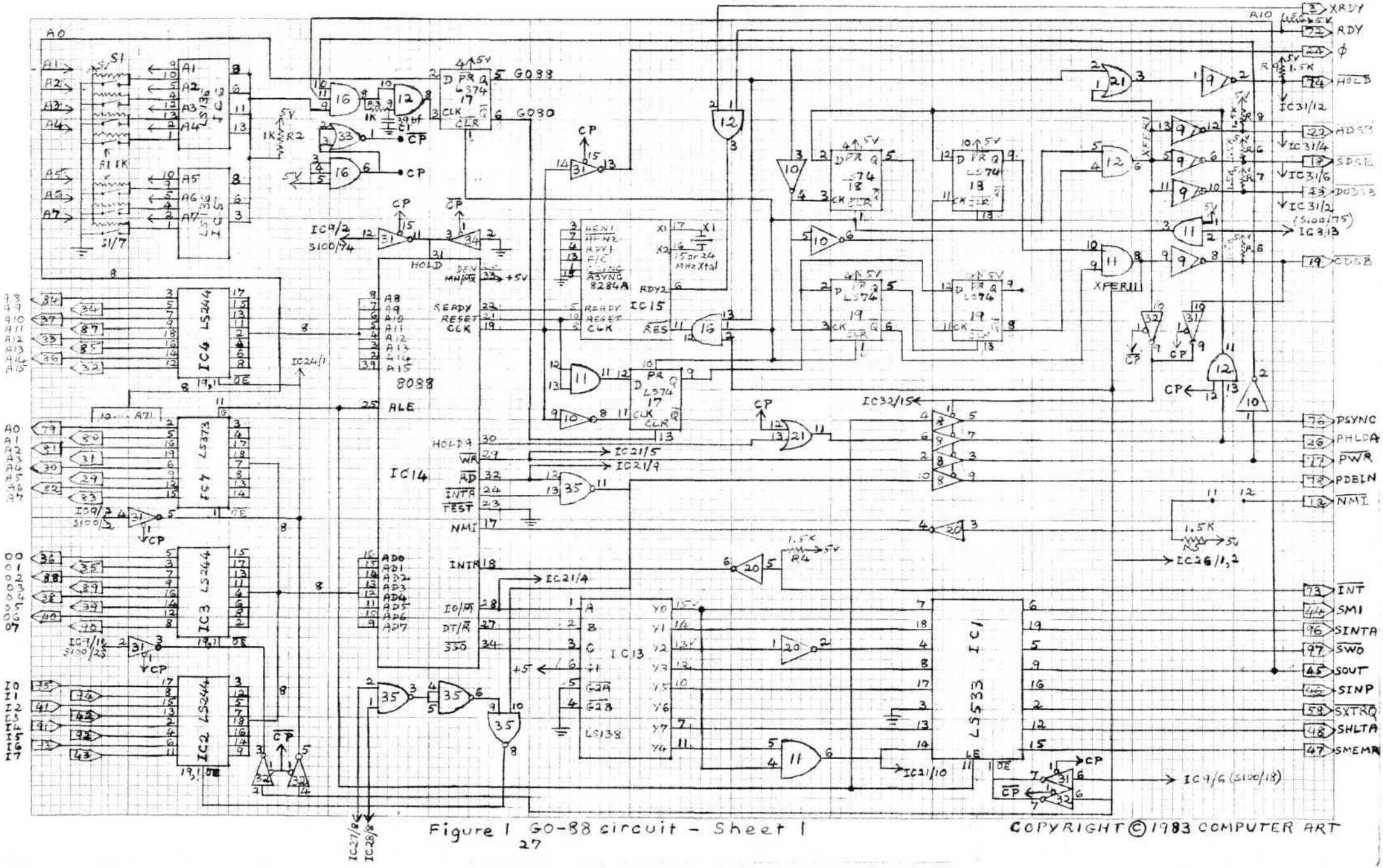

Figure 1 GO-88 circuit - Sheet 1

COPYRIGHT © 1983 COMPUTER ART

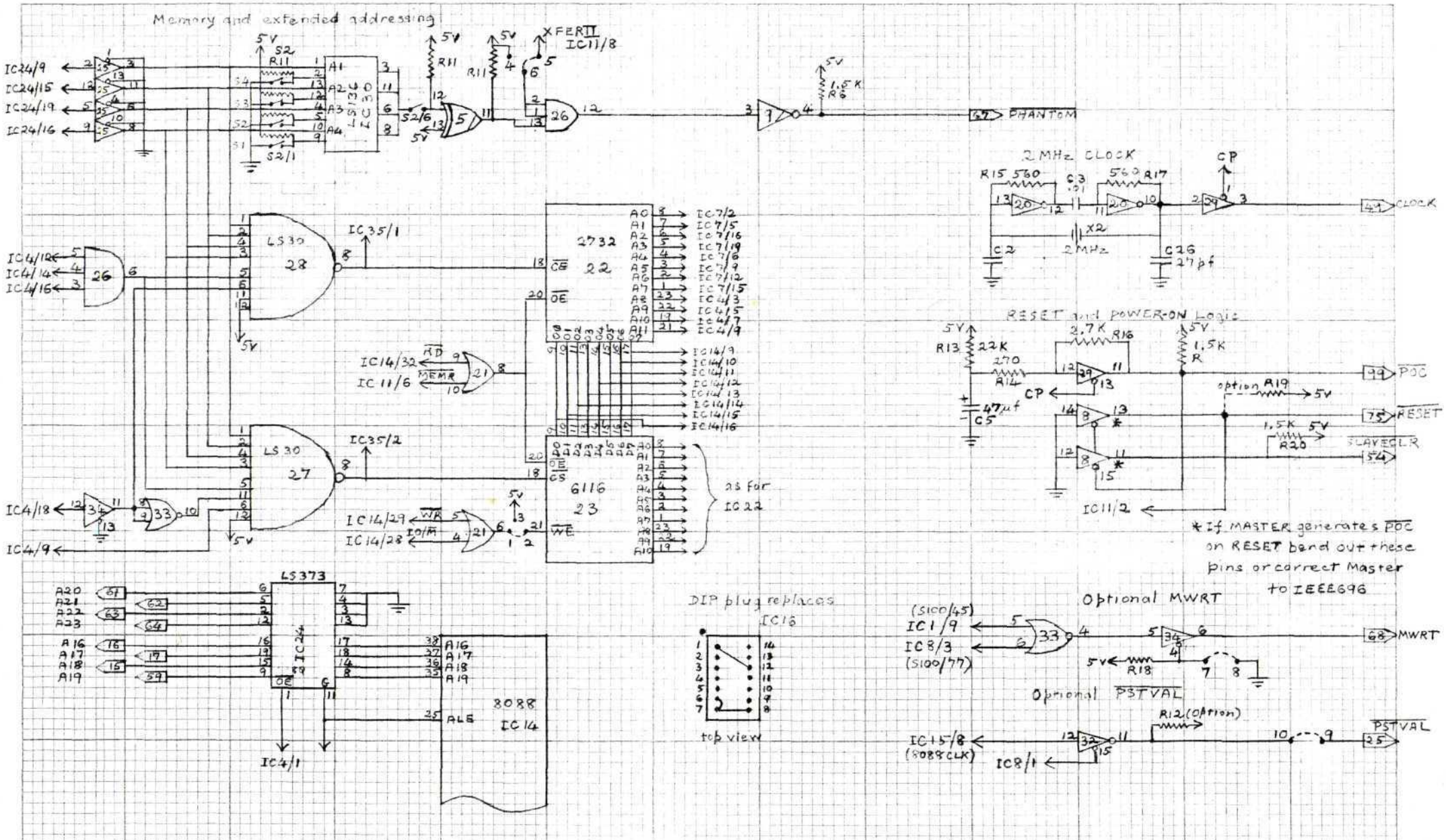

Figure 2 GO-88 circuit – Sheet 2

COPYRIGHT © 1983 COMPUTER ART

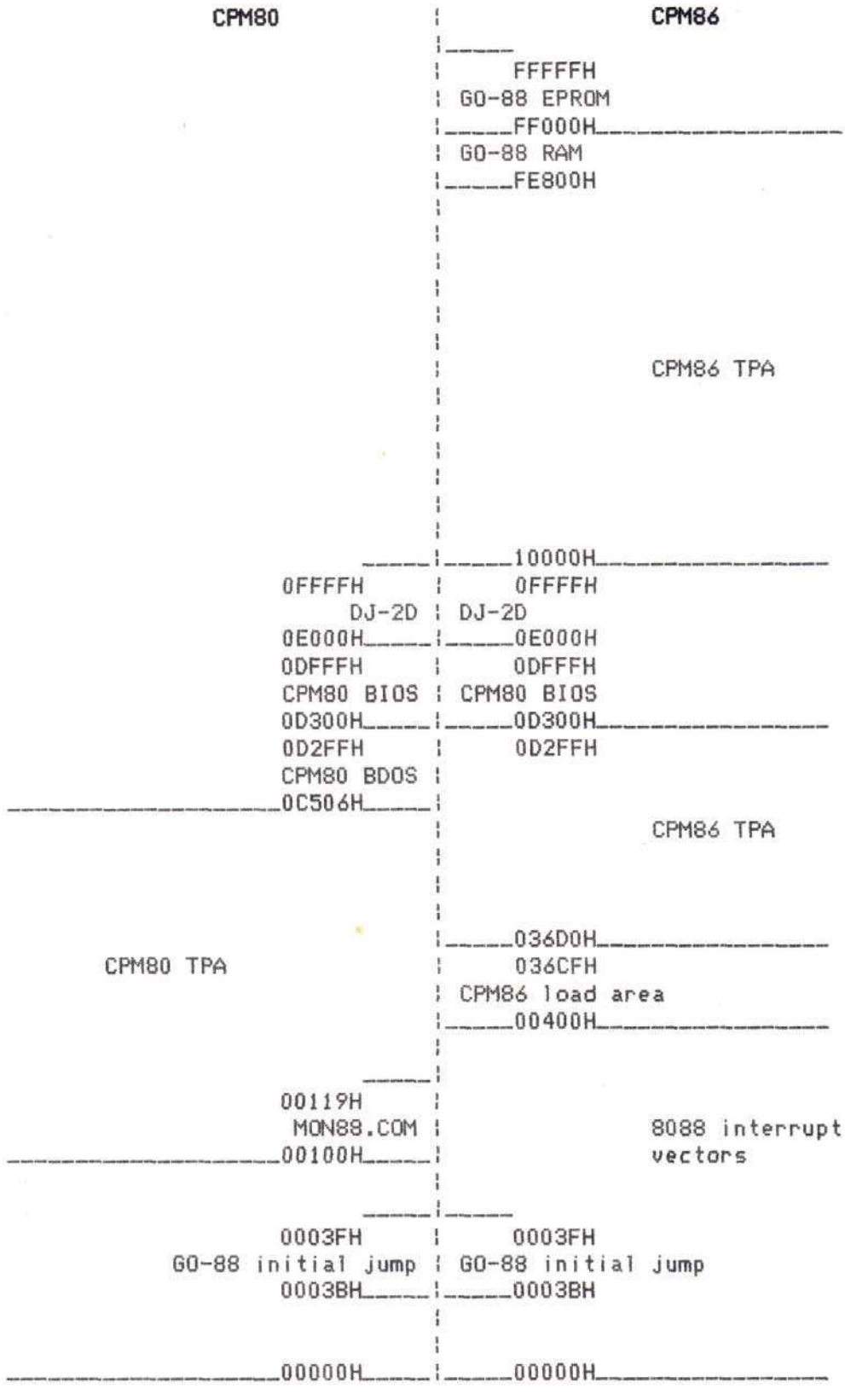

Figure 3 Memory Map - Model Coprocessor System

COMPUTER ART

GO-88 MANUAL

CPM86/DJ2D Supplement

This supplement contains the instructions for installing the full CPM86 operating system in a CPM80 S100 system containing GO-88 in the coprocessor mode and a Morrow DISCUS 2D floppy disk controller.

It is assumed that the S100 system is operating correctly as evidenced by normal operation of CPM80 and the ability to execute GO-88 monitor commands. It is also assumed that the model coprocessor system has been implemented as described in section 4 and section 5(1) of the manual.

The purpose of the procedure that follows is to prepare a disk file called CPM86.COM by joining the COMPUTER ART CB10S2D2 and INIT86 to your copy of CPM86. The procedure is similar in principle to that described in Digital Research's Adaptation procedures in their CPM86 System Guide. Actually it is simpler because we do not need a bootstrap loader.

The first thing to do is to prepare a disk formatted to the sector size you normally use and SYSGEN'd. Then PIP over to it the programs on the 8" Single density disk supplied by COMPUTER ART, together with the essential programs from your CPM86 distribution disk. On this disk you will also need PIP.COM and DDT.COM (or SID.COM or ZSID.COM). Insert this new disk in your A drive and make sure it is WRITE enabled.

```
Enter
PIP PARTCPM.H86 = CPM.H86,CBIOS2D2.H86

enter
GENCMD PARTCPM 8080 CODE[A40]

computer replies
BYTES READ 28BB
RECORDS WRITTEN 67

enter
ZSID (or SID or DDT)
IPARTCPM.CMD
R280

computer replies
NEXT 3700

enter
IINIT86.HEX
R

computer replies
NEXT 3700
```

Copyright (C) 1985 COMPUTER ART

As a check, you may now, if you wish,

enter
L0100
and
L293F

and you should see disassembled Z80 or 8080 code at those addresses.

Next, exit ZSID with CONTROL-C, and finally enter

SAVE 47 CPM86.COM

On entering CPM86 the computer should reply

CP/M-86 Version 1.1 for GO-88 in
coprocessor mode and DJ2D controller

Version 1.2 system generated 22/2/85

A(prompt)

Try loading DDT86 and looking at memory.

To return to CPM80 enter CPM80.

CPM86 can manage up to 8 non-contiguous areas of system memory and our BIOS contains the system memory segment table at 16-bit address 2FBEH. Digital Research's CPM86 literature explains the format of this table. As supplied, our BIOS assumes the target coprocessor system with three areas of memory available to CPM86 in which to load programs. If you have more RAM than that, you can customise the table using DDT. The reserve space in the table, for up to 5 more areas, is filled with zeros. Enter the new number of segments, starting paragraph address and lengths and SAVE as above.

Copyright (C) 1983 COMPUTER ART